Deep Clustering with Concrete K-Means

DEEP CLUSTERING WITH CONCRETE K-MEANS

Boyan Gao[1]*, Yongxin Yang*[1]*, Henry Gouk*[1]*, Timothy M. Hospedales*[1,2]

[1]School of Informatics, University of Edinburgh, United Kingdom
[2]Samsung AI Centre, Cambridge, United Kingdom

ABSTRACT

We address the problem of simultaneously learning a k-means clustering and deep feature representation from unlabelled data, which is of interest due to the potential for deep k-means to outperform traditional two-step feature extraction and shallow clustering strategies. We achieve this by developing a gradient estimator for the non-differentiable k-means objective via the Gumbel-Softmax reparameterisation trick. In contrast to previous attempts at deep clustering, our concrete k-means model can be optimised with respect to the canonical k-means objective and is easily trained end-to-end without resorting to time consuming alternating optimisation techniques. We demonstrate the efficacy of our method on standard clustering benchmarks.

***Index Terms*—** Deep Clustering, Unsupervised Learning, Gradient Estimation

1. INTRODUCTION

Clustering is a fundamental task in unsupervised machine learning with numerous applications. A key challenge for clustering is the inter-dependence between the data representation and the chosen distance metric. For example, the ubiquitous k-means algorithm assumes a fixed feature representation, distance metric, and existence of spherical clusters. These assumptions lead to poor performance if k-means is applied to complex high dimensional data such as raw image pixels, rather than higher level features. This observation motivates the vision of end-to-end deep clustering. Joint learning of data representation and k-means clustering has the potential to learn a "k-means friendly" space [1] in which high-dimensional data can be clustered without hand-engineering of feature representations. More generally, unifying unsupervised clustering and representation learning has the potential to alleviate the data annotation bottleneck in the standard supervised deep learning paradigm.

The main roadblock in deep k-means is the non-differentiability of the discrete cluster assignment in the objective function. Two recent methods—DEC [2] and DCN [1]—attempt to address this issue by proposing surrogate losses and alternating optimisation heuristics, respectively. However, the surrogate loss used by DEC may not lead to an optimal solution of the k-means objective. Furthermore it makes use of soft instance-cluster assignment, which is known to favour overlapping clusters compared to hard assignment methods [3], and more importantly does not provide the discrete assignments necessary for interpretability in some applications of k-means [3]. In contrast, DCN resorts to alternating optimisation rather than end-to-end gradient-based learning. This is slow and also restricts the ability to integrate clustering as a module in a larger backprop-driven deep network. In this paper we propose concrete k-means (CKM), the first end-to-end solution to optimising the true k-means objective jointly with representation learning. We achieve this by adapting the Gumbel-Softmax reparameterisation trick [4] to allow differentiation and backpropagation through the discrete cluster assignment. This CKM algorithm enables joint training of cluster centroids and representations, and can be optimised using standard methods for training deep networks. Furthermore, we show that CKM also provides a solver for shallow k-means, with comparable performance to the standard Lloyd's algorithm [5].

To summarise, our main contribution is the concrete k-means algorithm, the first method to enable joint end-to-end learning of clusters and representations in discrete-assignment deep k-means.

2. RELATED WORK

Clustering methods aim to find subgroups of data that are related according to some distance metric or notion of density. The performance of distance-based clustering algorithms is highly dependent on the data representation, and the goal of *deep* clustering is to learn a representation of the data that best facilitates clustering. k-means is perhaps the most ubiquitous clustering method [5, 6, 7], and it is widely used due to its simplicity and interpretability. For this reason, several attempts have been made to develop deep k-means generalisations. However, this is challenging due to the non-differentiable hard assignment of data points to cluster centres in the k-means objective. Xie et al. [2] show how to jointly optimise an autoencoder and a k-means model to get a "k-means friendly" latent space. The hard assignment in the k-means objective prevents them from optimising the true loss function, so their DEC method makes use of an approximation based on soft assignment of instances to clusters. However, this surrogate objective means that the solution to their model is not necessarily a minimum of the k-means objective. In contrast, DCN [1] resolves the issue by alternating optimisation. Each minibatch of

This work was supported by EPSRC grant EP/R026173/1, and NVIDIA Corporation GPU donation.

training data is first used to update the deep representation while keeping the centroids held constant, and then used to update the centroids while holding the representation constant. However, alternating optimisation may be slow and ineffective compared to an end-to-end solution. More importantly, it hampers integration of clustering as a module in a larger end-to-end deep learning system. We show how one can jointly train a deep representation and cluster centroids with the standard k-means objective using backpropagation and conventional deep learning optimisers.

3. CONCRETE K-MEANS

We first introduce the conventional k-means model. Following this, we show how to adapt the k-means objective to train cluster centroids and a deep neural network simultaneously. We refer to this novel generalisation as Concrete k-Means (CKM), due to the use of the concrete distribution [8].

3.1. Conventional k-Means

The k-means algorithm groups data points, $\{\vec{x}_i\}_{i=1}^N$ from some space, $\mathcal{X} \subset \mathbb{R}^d$, into k different clusters parameterised by centroids, $\{\vec{\mu}_i\}_{i=1}^k$, also from $\mathbb{R}^d$. By stacking each $\vec{\mu}_i$, the centroids can be collectively represented as a matrix, $M \in \mathbb{R}^{k\times d}$, where each row corresponds to a cluster centre. The k-means objective is to find the assignment and set of centroids that minimise the distance between each point and its associated centroid.

$$\min_{H,M} \|X - HM\|_F^2 \qquad (1)$$
$$\text{s.t.} \|\vec{h}_j\|_1 = 1, H \in \{0,1\}^{N\times k}$$

where $H \in \{0,1\}^{N\times k}$ is a binary matrix that represents the cluster assignments of each point, $\vec{h}_j$ is the jth row of H, and N is number of data points.

The most common method for learning k-means clusters is Lloyd's algorithm [5], which can be formalised as an alternating optimisation problem. The first step is to find the optimal cluster assignments given the current cluster centres, and the second step is to find the optimal cluster centres for a fixed set of assignments. Both of these optimisation problems permit closed form solutions: the first can be solved by finding the cluster centre closest (according to Euclidean distance) to each data point, and the second is minimised when each cluster centre is set to the mean of its assigned data points. Lloyd's algorithm alternates between finding locally optimal solutions to these two problems until the cluster assignments become stable.

3.2. Deep k-Means with Concrete Gradients

Deep k-means strategies aim to cluster the data in a learned embedding space $\mathcal{Z}$ rather than the raw input space $\mathcal{X}$. The embedding is defined via a learned neural network $\mathbf{z} = f_{\vec{\phi}}(\mathbf{x})$. Following previous work [2, 1], we avoid degenerate solutions by defining an autoencoder that regularizes the latent space by reconstructing the original input. Specifically, we define an encoder, $f_{\vec{\phi}} : \mathcal{X} \to \mathcal{Z}$, which maps from the input space to the latent space, and decoder, $g_{\vec{\varphi}} : \mathcal{Z} \to \mathcal{X}$, which maps from the latent space back to the input space. These networks are then composed and their parameters, $\vec{\phi}$ and $\vec{\varphi}$, are trained to minimise the reconstruction error,

$$\mathcal{L}^{AE}(X, \vec{\phi}, \vec{\varphi}) = \sum_{i=1}^{N} \|\vec{x}_i - g_{\vec{\varphi}}(f_{\vec{\phi}}(\vec{x}_i))\|_2^2. \qquad (2)$$

The proposed algorithm performs clustering in the latent space $\mathcal{Z}$ rather than the input space $\mathcal{X}$. In the conventional k-means algorithm, a data point is assigned to the cluster with the nearest centroid, as measured by Euclidean distance. The hard assignment operation is not differentiable, thus precluding the direct use of standard gradient-based optimisation techniques for training neural networks. We reformulate the non-differentiable assignment operation using the Straight-Through Gumbel-Softmax estimator [4]. This reparameterisation trick enables the use of a probabilistic hard assignment during the forward propagation, while also allowing gradients to be backpropagated through a soft assignment in order to train the network. We keep the Euclidean distance of the traditional k-means algorithm, and model cluster assignment probabilities using normalised radial basis functions (RBFs),

$$p(C_{i,j}|\vec{x}_i) = \frac{\exp\{-\sigma^{-2}\|\vec{z}_i - \vec{\mu}_j\|_2^2\}}{\sum_{c=1}^{k} \exp\{-\sigma^{-2}\|\vec{z}_i - \vec{\mu}_c\|_2^2\}}, \qquad (3)$$

where $C_{i,j}$ is the event that instance i is assigned to cluster j, and $\vec{z}_i = f_{\vec{\phi}}(\vec{x}_i)$, and we have omitted the dependence of p on M and $\vec{\phi}$ to keep notation compact. One can now sample one-hot vectors from $p(C_i|\vec{x}_i)$, while simultaneously being able to backpropagate through the sampling process, by instead sampling from a Gumbel-Softmax distribution—a continuous relaxation of the distribution of one-hot encoded samples from $p(C_i|\vec{x}_i)$. By introducing Gumbel distributed random variables, G, one can make use of a reparameterisation trick to sample from the Gumbel-Softmax distribution,

$$h_{i,j} = \frac{\exp\{\tau^{-1}(\log(p(C_{i,j}|\vec{x}_i)) + G_j)\}}{\sum_{c=1}^{k} \exp\{\tau^{-1}(\log(p(C_{i,c}|\vec{x}_i)) + G_c)\}}, \qquad (4)$$

where $h_{i,j}$ is the jth component of the vector, $\vec{h}_i$ corresponding to instance $\vec{x}_i$, and $\tau \in (0, \infty)$ is a temperature hyperparameter used for controlling the entropy of the continuous relaxation. As τ goes to zero, $\vec{h}_i$ converges towards true one-hot samples from $p(C_i|\vec{x}_i)$. In contrast, as τ goes to infinity, the $\vec{h}_i$ converge towards a uniform distribution. In practice, we start training with a high temperature and gradually anneal it towards zero as training progresses. The $\vec{h}_i$ vectors can be discretised by rounding the largest component to one, and all others to zero, giving a truly discrete sample distributed according to $p(C_i|\vec{x}_i)$. We denote the discretization of $\vec{h}_i$ by $\tilde{\vec{h}}_i$. With this notation, we define the concrete k-means loss as

$$\mathcal{L}^{CKM}(X, M, \vec{\phi}) = \sum_{i=1}^{N} ||f_{\vec{\phi}}(\vec{x}_i) - \tilde{\vec{h}}_i M||_2^2, \qquad (5)$$

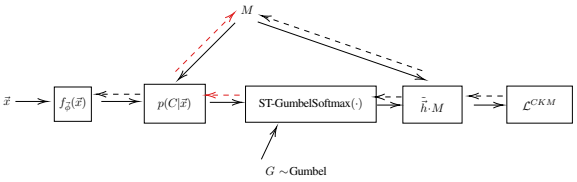

Fig. 1. A computational graph view of the information flow for the concrete k-means algorithm. Solid arrows indicate computation during forward propagation, and dashed arrows indicate gradient flow during backpropagation. The red dashed arrows show which gradients are computed by the concrete gradient estimator.

Algorithm 1: Concrete k-means clustering

Input: $X, \alpha, \eta, \lambda, \vec{\phi}^{(0)}, \vec{\varphi}^{(0)}, M^{(T_1)}$

Output: $\vec{\phi}^{(T_2)}, \vec{\varphi}^{(T_2)}, M^{(T_2)}$

begin

  for $t \leftarrow 0$ **to** T_1 **do**

    $(\vec{\phi}^{(t+1)}, \vec{\varphi}^{(t+1)}) \leftarrow (\vec{\phi}^{(t)}, \vec{\varphi}^{(t)}) - \alpha \nabla_{(\vec{\phi},\vec{\varphi})} \mathcal{L}^{AE}(X)$;

  end

  for $t \leftarrow T_1$ **to** T_2 **do**

    $\vec{\phi}^{(t+1)} \leftarrow \vec{\phi}^{(t)} - \eta \nabla_{\vec{\phi}} (\mathcal{L}^{AE}(X) + \lambda_1 \mathcal{L}^{CKM}(X))$;

    $\vec{\varphi}^{(t+1)} \leftarrow \vec{\varphi}^{(t)} - \eta \nabla_{\vec{\varphi}} \mathcal{L}^{AE}(X)$;

    $M^{(t+1)} \leftarrow M^{(t)} - \eta \nabla_{M} \lambda_1 \mathcal{L}^{CKM}(X)$;

  end

end

noting that $\tilde{\vec{h}}_i$ is a row vector. During the forward propagation, $\tilde{\vec{h}}_i$ is used for evaluating the k-means loss. During the backward pass, the gradient is estimated by back-propagating though the same loss, but parameterised by $\vec{h}_i$ instead of $\tilde{\vec{h}}_i$. This method of computing gradients for one-hot encoded categorical variables is known as the straight through Gumbel-softmax estimator [4], or the concrete estimator [8].

To train our Concrete k-means, we optimize the main CKM objective in Eq. 5 along with the autoencoder, with respect to encoder and decoder parameters as well as cluster centres. The full objective is:

$$\min_{M,\vec{\phi},\vec{\varphi}} \mathcal{L}^{AE}(X,\vec{\phi},\vec{\varphi}) + \lambda_1 \mathcal{L}^{CKM}(X,M,\vec{\phi}), \tag{6}$$

where λ_1 is a regularisation strength hyperparameter. The stochastic computational graph [9] in Figure 1 illustrates the flow of information during training for both the forward and backward passes.

3.3. Shallow Concrete k-means

Our algorithm is motivated by the vision of joint clustering and representation learning. Nevertheless, it is worth noting that as a byproduct it provides a novel optimisation strategy for the conventional k-means objective in Equation 1. We simply run CKM on raw features, which can be interpreted as fixing the encoder and decoder to the identity function, and solve Equation 6 for centroids M alone. Thus we use stochastically estimated gradients to solve conventional k-means by gradient descent rather than alternating minimisation [5].

4. EXPERIMENTS

In this section, we evaluate CKM in conventional shallow and deep clustering.

4.1. Shallow Clustering

The concrete k-means method presented in Section 3.2 does not require the presence of a feature extraction network, and can thus be used to optimise the k-means objective in the 'shallow' setting where Lloyd's [5] and k-means++ [6] are typically applied. Our first experiment aims to confirm if the CKM gradient-based stochastic optimisation matches the performance of the standard k-means solvers. Table 2 reports the clustering results of our shallow CKM and sklearn's k-means++ implementation on ten UCI datasets. The evaluation metrics used for these experiments are normalized mutual information (NMI) [10], adjusted rand index (ARI) [11], and cluster purity (ACC). The values of ACC and NMI are rescaled to lie between zero and one, with higher values indicating better performance. The range of the ARI is negative one to one. We can see that CKM performs comparably to the industry standard k-means optimizer.

4.2. Deep Clustering

Datasets We conduct deep clustering experiments are using the following datasets from the image and natural language domains: **MNIST** [12] consists of 70,000 greyscale images of handwritten digits. There are 10 classes and each image is 28×28 pixels, with the digits appearing inside the central 20×20 pixel area. **USPS** is a dataset of 16×16 pixel handwritten digit images. The first 7,291 images are designated as the training fold, and the remaining 2,007 are used for evaluating the final performance of the models. **20Newsgroups** was generated by collecting a total of 18,846 posts over 20 different newsgroups. We use the same preprocessing as [1], where the tf-idf representation of the 2,000 most frequently occurring words are used as features.

Metrics The evaluation metrics used for these experiments are normalized mutual information (NMI) [10], adjusted rand index (ARI) [11], and cluster purity (ACC). The values of ACC and NMI are rescaled to lie between zero and one, with higher values indicating better performance. The range of the ARI is negative one to one.

Architecture Like most deep clustering methods (e.g., [1] and [2]), our approach involves pretraining an autoencoder before optimising the clustering objective. The encoder architecture used for the clustering experiments on MNIST and USPS contains four fully connected layers with 500, 500, 2000, and 10 units,

Method	MNIST			USPS			20NEWSGROUP		
	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
KM^h	51.8±0.4	36.5±0.4	53.3±0.5	60.0± 0.7	44.0 ±0.9	58.0 ±0.9	22.7±1.7	8.0±1.4	22.6±2.1
AE+KM^h	74.3±0.9	66.9±0.8	80.6±1.2	68.1±0.3	59.4±0.4	68.4±0.7	42.0±1.7	28.3±1.2	44.3±2.3
DCN^h	**81.7±1.1**	75.2±1.2	83.1±1.9	71.9±1.2	61.9±1.4	**73.9±0.8**	44.7±1.5	34.4±1.3	46.3±2.9
DEC^e	80.4±1.3	76.3±1.8	84.2±1.7	**72.6±1.1**	**63.8±0.9**	71.1±2.5	**48.6±1.2**	**35.4±1.4**	**49.1±2.5**
CKM^{h,e}	81.4±1.8	**77.7±1.1**	**85.4±2.1**	70.7±0.2	61.3±0.2	72.1±0.4	46.5±1.4	34.1±1.6	47.3±2.3

Table 1. Deep clustering results on MNIST, USPS and 20 Newsgroups. ^{h} indicates methods with interpretable hard assignments, and ^{e} indicates methods with end-to-end learning by backpropagation. Only our Concrete k-means combines hard assignment and end-to-end learning. Blue indicates best result among hard clustering methods, bold indicates best result overall.

	Shallow CKM			k-means		
	NMI	ARI	ACC	NMI	ARI	ACC
pendigits	0.50±0.04	0.33±0.05	0.49±0.05	0.51±0.04	0.34±0.05	0.49±0.05
dig44	0.33±0.04	0.20±0.05	0.40±0.05	0.33±0.04	0.20±0.05	0.40±0.05
vehicle	0.15±0.03	0.09±0.03	0.40±0.03	0.15±0.03	0.09±0.03	0.40±0.03
letter	0.35±0.01	0.13±0.01	0.26±0.01	0.35±0.01	0.13±0.01	0.25±0.01
segment	0.41±0.05	0.27±0.05	0.46±0.04	0.41±0.05	0.27±0.05	0.46±0.04
waveform	0.35±0.04	0.27±0.04	0.57±0.05	0.36±0.01	0.25±0.01	0.52±0.02
vowel	0.41±0.01	0.21±0.01	0.36±0.02	0.42±0.01	0.21±0.01	0.36±0.02
spambase	0.10±0.03	0.09±0.05	0.66±0.04	0.10±0.03	0.09±0.05	0.66±0.04
twonorm	0.84±0.00	0.91±0.00	0.98±0.00	0.84±0.01	0.91±0.01	0.98±0.00
sat	0.58±0.05	0.48±0.08	0.64±0.07	0.58±0.05	0.48±0.08	0.64±0.07

Table 2. Shallow CKM uses gradient estimation to solve the standard fixed-feature k-means problem equally well to the conventional alternating minimisation based k-means++ [6] implemented in scikit-learn.

respectively. For the 20Newsgroup experiments, the smaller encoder with 250, 100, and 20 units described by [1] is used. The decoder that maps the hidden representation back to the input space is the mirror version of the encoder.

Competitors Comparisons are made with **DEC** [2] and **DCN** [1], as well as some simple baselines. **KM** applies classic shallow K-means to raw input features from $\mathcal{X}$. **AE+KM** performs two step dimensionality reduction and clustering by training an autoencoder with the same architecture as CKM to embed instances into the latent space $\mathcal{Z}$, and then fixes this space before applying classic k-means clustering.

In Table 1 we illustrate the results of our CKM and the comparison with other models. For all experiments in this section, k is set to the number of classes present in the dataset. We run each method 15 times with different initial random seeds and report the mean and standard deviation of each result.

From the results, we can see that all the deep methods outperform shallow k-means on raw-features (KM), and furthermore all the jointly trained methods outperform the two-step baseline (AE+KM). Compared to the published state of the art methods, our CKM approach generally performs best on MNIST, and comparably to DEC and DCN on USPS and 20Newsgroup. Importantly, our CKM is the only high-performing method to combine the favorable properties of hard-assignment, which is important for interpretability in many applications [3]; and end-to-end deep learning, which is important to be able to integrate clustering functionality as a module into a larger backpropagation-driven system.

Runtime Efficiency: Comparing the three deep clustering methods, DCN's alternating optimisation is slower than the end-to-end DEC and CKM. For MNIST, the clock time per epoch is $11s$, $10s$, and $36s$ for CKM, DEC and DCN respectively.

5. CONCLUSION

This paper proposes the concrete k-means deep clustering framework. Our stochastic hard assignment method is able to estimate gradients of the nondifferentiable k-means loss function with respect to cluster centres. This, in turn, enables end-to-end training of a neural network feature extractor and a set of cluster centroids in this latent space. Our experimental results show that the proposed method is competitive with state-of-the-art approaches for solving deep clustering problems.

6. REFERENCES

[1] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[2] Junyuan Xie, Ross Girshick, and Ali Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[3] Michael Kearns, Yishay Mansour, and Andrew Y. Ng, "An information-theoretic analysis of hard and soft assignment methods for clustering," in *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 1997.

[4] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical reparameterization with gumbel-softmax," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[5] Stuart Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[6] David Arthur and Sergei Vassilvitskii, "K-means++: The advantages of careful seeding," in *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[7] Jason Xu and Kenneth Lange, "Power k-means clustering," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[8] Chris J Maddison, Andriy Mnih, and Yee Whye Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[9] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel, "Gradient estimation using stochastic computation graphs," in *Advances in Neural Information Processing Systems 29*, 2015.

[10] Deng Cai, Xiaofei He, and Jiawei Han, "Locally consistent concept factorization for document clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 902–913, 2010.

[11] Ka Yee Yeung and Walter L Ruzzo, "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.